Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** necessari alla compilazione e alla corretta esecuzione del programma.
Nota: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire sempre *"Rebuild All"*.

Una azienda fornisce i pasti per degli studenti universitari. Gli studenti, all'atto di pagare, passano una card studenti in un lettore apposito, l'impiegata segna sul registratore di cassa cosa hanno preso, e il tutto viene scritto in un file di testo. In particolare, sul file "**`cassa.txt`**" vengono scritte le seguenti informazioni: **l'id** unico della card studenti (un intero); a seguire, separati da uno spazio, una o più delle seguenti stringhe: "**`primo`**", "**`secondo`**", "**`frutta`**", "**`contorno`**", "**`dolce`**". Il significato inteso è che uno studente che abbia preso ad esempio primo, secondo e frutta, vedrà comparire nel file di testo le tre stringhe corrispondenti. Ovviamente alcuni studenti prenderanno più cose, altri studenti meno cose; inoltre alcuni studenti prenderanno più volte la stessa cosa: ad esempio uno studente che prenderà due volte il secondo vedrà registrata due volte la stringa "**`secondo`**".
Si noti che nell'ambito del singolo pasto le stringhe possono comparire in ordine qualsiasi; inoltre i pasti sono segnati in maniera confusa nel file: a volte un pasto per riga, a volte invece più pasti sulla stessa riga. In ogni caso il sistema inserisce sempre una stringa speciale "@@@" di separazione tra un pasto e il successivo.
In un secondo file di nome "**`tariffe.txt`**" si tiene memoria delle tariffe: su ogni riga, per ogni tipo di cibo (primo, secondo, etc.) si memorizza il costo. Quindi su ogni riga si trova: una stringa con il nome di una pietanza; e a seguire, separato da uno spazio, il costo (un float).

Esercizio 1 – Strutture dati Pasto, Tariffa, e funzioni di lett./scritt. (mod. element.h e cassa.h/.c)

Si definisca una opportuna struttura dati **`Pasto`** per memorizzare un singolo pasto, cioè l'id della carta studente, e il numero di primi, di secondi, di frutta, di contorni e di dolci presi in quel pasto (quindi cinque interi indicanti cinque quantità). Si definisca poi una opportuna struttura dati **`Tariffa`**, per la memorizzazione del costo di una singola pietanza: quindi una stringa per la pietanza, e un float per il costo.

Si definisca la funzione:

```
Pasto * leggiPasti(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file, legga da tale file i dati di tutti i pasti e li restituisca tramite un array (allocato dinamicamente e della dimensione minima) di strutture dati di tipo **`Pasto`**. Tramite il parametro dim, la funzione dovrà restituire la dimensione dell'array allocato dinamicamente. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a NULL e dim pari a zero.
Si definisca la procedura RICORSIVA:

```
void stampaPasti(Pasto * elenco, int dim);
```

che, ricevuto in ingresso un puntatore ad un array di strutture dati di tipo **`Pasto`**, e la sua dimensione dim, stampi in modo RICORSIVO il contenuto dell'array.

Si definisca la funzione:

```
list leggiTariffe(char * fileName);
```

che, ricevuto in ingresso il nome di un file, legga da tale file i dati relativi alle tariffe e li restituisca tramite una lista di strutture dati di tipo **`Tariffa`**. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, e restituire una lista vuota.
Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

Esercizio 2 –Ordinamento ed eliminazione ripetuti (moduli element.h/.c e cassa.h/.c)

Il candidato definisca una procedura:

```
void ordina(Pasto * v, int dim);
```

che, ricevuti in ingresso un vettore di strutture dati di tipo **`Pasto`** e la dimensione di tale vettore, ordini il vettore secondo il seguente criterio: in ordine crescente in base all'id della carta studente; a parità di id, in ordine crescente in base al numero di primi consumati; a parità, in ordine crescente in base al numero di secondi consumati; a parità ulteriore, in ordine crescente in base al numero di dolci consumati. A tal scopo, il candidato utilizzi l'algoritmo di ordinamento "merge sort" visto a lezione.

Il file delle tariffe è aggiornato dal management dell'azienda fornitrice dei pasti. Quando aggiornano le tariffe, i manager si limitano ad aggiungere in una riga qualunque del file il nome della pietanza (primo, secondo etc.) e il nuovo prezzo. Così facendo, dopo poco tempo nel file si trovano le pietanze duplicate, ma con prezzi diversi. È quindi necessario filtrare la lista dei prezzi, ed eliminare i duplicati. Il management ha deciso che nella lista ripulita si devono tenere le tariffe coi prezzi più alti. Si definisca quindi una funzione:

```
list filtra(list elenco);
```

che, ricevuto in ingresso una lista di strutture dati di tipo **`Tariffa`**, restituisca in uscita una nuova lista contenete le tariffe senza duplicati. In caso di tariffe duplicate, si deve sempre tenere la tariffa dal prezzo più alto.

Esercizio 3 – Calcolo spese (modulo cassa.h/cassa.c)

Si sviluppi una funzione:

```
float spesa(Pasto * pasti, int dim, list tariffe, int id);
```

che, ricevuti in ingresso un array di strutture dati di tipo **`Pasto`** e la dimensione di tale array, la lista delle tariffe (senza duplicati), e l'identificatore di una specifica carta studente, restituisca la spesa totale attribuibile alla carta studente specificata, relativamente a tutti i pasti memorizzati nell'array (relativamente ovviamente all'id specificato). Attenzione: la spesa di ogni singolo pasto è data dalla somma dei costi delle singole pietanze: è poi ovvio che se uno studente ha preso due primi, ad esempio, dovrà pagare per due primi.

Esercizio 4 Richiesta dell'id di una carta studente, stampa della spesa, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che, dopo aver letto dai file forniti nello StartKit i dati, chieda all'utente di specificare l'id di una carta studente, e stampi a video il totale dovuto da quello studente.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

"element.h":

```
#define _USE_CRT_NO_WARNINGS

#ifndef _ELEMENT_H
#define _ELEMENT_H

#define DIM_STRINGA 256

typedef struct {
	int id;
	int primi;
	int secondi;
	int frutta;
	int contorni;
	int dolci;
} Pasto;

typedef struct {
	char nome[9];
	float prezzo;
} Tariffa;

typedef Tariffa element;

int compare(Pasto p1, Pasto p2);

#endif
```

"element.c":

```
#include "element.h"

int compare(Pasto p1, Pasto p2) {
	int result;

	result = p1.id - p2.id;
	if (result == 0)
		result = p1.primi - p2.primi;
	if (result == 0)
		result = p1.secondi - p2.secondi;
	if (result == 0)
		result = p1.dolci - p2.dolci;
	return result;
}
```

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

"list.h"

```
#ifndef LIST_H
#define LIST_H

#include "element.h"

typedef struct list_element
{
    element value;
    struct list_element  *next;
} item;

typedef  item* list;

typedef  int boolean;

/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);

void showlist(list l);
void freelist(list l);
int member(element el, list l);

#endif
```

"list.c":

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "list.h"

/* OPERAZIONI PRIMITIVE */
list  emptylist(void)            /* costruttore lista vuota */
{
    return NULL;
}

boolean  empty(list l)     /* verifica se lista vuota */
{
    return (l==NULL);
}

list  cons(element e, list l)
{
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
```

```
  return(t);
}

element head(list l) /* selettore testa lista */
{
  if (empty(l)) exit(-2);
  else return (l->value);
}

list  tail(list l)         /* selettore coda lista */
{
  if (empty(l)) exit(-1);
  else return (l->next);
}


void showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%s %f\n",
                     temp.nome, temp.prezzo);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}



int member(element el, list l) {
       int result = 0;
       while (!empty(l) && !result) {
              result = (strcmp(el.nome, head(l).nome) == 0);
              if (!result)
                     l = tail(l);
       }
       return result;
}




void freelist(list l) {
       if (empty(l))
              return;
       else {
              freelist(tail(l));
              free(l);
       }
       return;
}
```

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

"cassa.h":

```
#define _CRT_SECURE_NO_WARNINGS
#ifndef _PASTI_H
#define _PASTI_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "element.h"
#include "list.h"

// Es. 1
Pasto * leggiPasti(char * fileName, int * dim);
void stampaPasti(Pasto * elenco, int dim);
list leggiTariffe(char * fileName);

// Es. 2
void ordina(Pasto * v, int dim);
list filtra(list elenco);

// Es. 3
float spesa(Pasto * pasti, int dim, list tariffe, int id);

#endif
```

"cassa.c":

```
#include "cassa.h"
// Es. 1
void init(Pasto * p) {
	p->primi = 0;
	p->secondi = 0;
	p->dolci = 0;
	p->frutta = 0;
	p->contorni = 0;
}

Pasto * leggiPasti(char * fileName, int * dim) {
	FILE * fp;
	Pasto * result;
	Pasto temp;
	char cibo[DIM_STRINGA];
	int finePasto;
	char ch;
	int cont;
	init(&temp);
	result = NULL;
	*dim = 0;
	cont = 0;
	fp = fopen(fileName, "rt");
	if (fp == NULL) {
		printf("Errore nell'apertura del file: %s\n", fileName);
	}
	else {
		// conto quanti pasti ci sono nel file
		while (fscanf(fp, "%d", &(temp.id)) == 1) {
```

```
                init(&temp);
                finePasto = 0;
                while (!finePasto && fscanf(fp, "%s", cibo)) {
                        if (strcmp("@@@", cibo) == 0) {
                                finePasto = 1;
                                cont++;
                        }
                }
        }
        // alloco memoria e riavvolgo il file
        result = (Pasto *)malloc(cont * sizeof(Pasto));
        rewind(fp);
        // procedo alla lettura e copio in memoria
        while (fscanf(fp, "%d", &(temp.id)) == 1 && *dim<cont) {
                init(&temp);
                finePasto = 0;
                while (!finePasto && fscanf(fp, "%s", cibo)) {
                        if (strcmp("primo", cibo) == 0) temp.primi++;
                        if (strcmp("secondo", cibo) == 0) temp.secondi++;
                        if (strcmp("frutta", cibo) == 0) temp.frutta++;
                        if (strcmp("contorno", cibo) == 0) temp.contorni++;
                        if (strcmp("dolce", cibo) == 0) temp.dolci++;
                        if (strcmp("@@@", cibo) == 0) finePasto = 1;
                        if (finePasto) {
                                result[*dim] = temp;
                                *dim = *dim + 1;
                        }
                }
        }
        fclose(fp);
    }
    return result;
}

void stampaPasti(Pasto * elenco, int dim) {
    if (dim == 0)
        return;
    else {
        stampaPasti(elenco, dim - 1);
        printf("ID: %d\n\tprimi: %d", elenco[dim-1].id, elenco[dim-1].primi);
        printf("\tsecondi: %d\n\tfrutta: %d", elenco[dim-1].secondi, elenco[dim -
1].frutta);
        printf("\tcontorni: %d\n\tdolce: %d\n\n", elenco[dim - 1].contorni,
elenco[dim - 1].dolci);
    }
}

list leggiTariffe(char * fileName) {
    FILE * fp;
    list result;
    Tariffa temp;
    result = emptylist();
    fp = fopen(fileName, "rt");
    if (fp == NULL) {
        printf("Errore nell'apeertura del file %s\n", fileName);
    }
    else {
```

```
		while (fscanf(fp, "%s%f", temp.nome, &(temp.prezzo)) == 2) {
			result = cons(temp, result);
		}
		fclose(fp);
	}
	return result;
}

// Es. 2
void  merge(Pasto v[], int i1, int i2,  int fine, Pasto vout[]) {
	int i = i1, j = i2, k = i1;
	while (i <= i2 - 1 && j <= fine) {
		if (compare(v[i], v[j])<0)
			vout[k] = v[i++];
		else
			vout[k] = v[j++];
		k++;
	}
	while (i <= i2 - 1) {
		vout[k] = v[i++]; k++;
	}
	while (j <= fine) {
		vout[k] = v[j++]; k++;
	}
	for (i = i1; i <= fine; i++)
		v[i] = vout[i];
}
void mergeSort(Pasto v[], int first, int last, Pasto vout[]) {
	int mid;
	if (first < last) {
		mid = (last + first) / 2;
		mergeSort(v, first, mid, vout);
		mergeSort(v, mid + 1, last, vout);
		merge(v, first, mid + 1, last, vout);
	}
}
void ordina(Pasto * v, int dim) {
	Pasto * temp;
	temp = (Pasto *)malloc(sizeof(Pasto) * dim);
	mergeSort(v, 0, dim - 1, temp);
	free(temp);
}

Tariffa trovaMax(list elenco, char * nome) {
	Tariffa result;
	Tariffa temp;
	strcpy(result.nome, nome);
	result.prezzo = 0.0F;
	while (!empty(elenco)) {
		temp = head(elenco);
		if (strcmp(nome, temp.nome) == 0 && temp.prezzo > result.prezzo)
			result = temp;
		elenco = tail(elenco);
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

```
list filtra(list elenco) {
	list result;
	list temp;
	Tariffa current;
	temp = elenco;
	result = emptylist();
	while (!empty(temp)) {
		current = head(temp);
		if (!member(current, result)) {
			result = cons(trovaMax(elenco, current.nome), result);
		}
		temp = tail(temp);
	}

	return result;
}

float spesa(Pasto * pasti, int dim, list tariffe, int id) {
	int i;
	int trovato;
	list temp;
	float totale;
	Tariffa taff;
	totale = 0.0F;
	for (i = 0; i < dim; i++) {
		if (pasti[i].id == id) {
			temp = tariffe;
			while (!empty(temp)) {
				taff = head(temp);
				if (strcmp("primo", taff.nome) == 0) totale = totale +
taff.prezzo*pasti[i].primi;
				if (strcmp("secondo", taff.nome) == 0) totale = totale +
taff.prezzo*pasti[i].secondi;
				if (strcmp("contorno", taff.nome) == 0) totale = totale +
taff.prezzo*pasti[i].contorni;
				if (strcmp("frutta", taff.nome) == 0) totale = totale +
taff.prezzo*pasti[i].frutta;
				if (strcmp("dolce", taff.nome) == 0) totale = totale +
taff.prezzo*pasti[i].dolci;
				temp = tail(temp);
			}
		}
	}
	return totale;
}
```

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

"main.c":

```
#define _CRT_SECURE_NO_WARNINGS
#include "element.h"
#include "cassa.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
	{ // es.1
		Pasto * elenco;
		int dim;
		list tariffe;
		elenco = leggiPasti("cassa.txt", &dim);
		stampaPasti(elenco, dim);
		tariffe = leggiTariffe("tariffe.txt");
		showlist(tariffe);
		free(elenco);
		freelist(tariffe);
	}
	{ // es. 2
		Pasto * elenco;
		int dim;
		list tariffe;
		list tariffeFiltrate;
		elenco = leggiPasti("cassa.txt", &dim);
		ordina(elenco, dim);
		stampaPasti(elenco, dim);
		tariffe = leggiTariffe("tariffe.txt");
		tariffeFiltrate = filtra(tariffe);
		showlist(tariffeFiltrate);
		free(elenco);
		freelist(tariffe);
		freelist(tariffeFiltrate);
	}
	{ // es. 3 & 4
		Pasto * elenco;
		int dim;
		list tariffe;
		list tariffeFiltrate;
		int id;
		elenco = leggiPasti("cassa.txt", &dim);
		tariffe = leggiTariffe("tariffe.txt");
		tariffeFiltrate = filtra(tariffe);
		printf("Specificare l'id di cui calcolare il costo: ");
		scanf("%d", &id);
		printf("il costo totale e': %6.2f\n", spesa(elenco, dim, tariffeFiltrate,
id));
		free(elenco);
		freelist(tariffe);
		freelist(tariffeFiltrate);
	}
	return 0;
}
```

Fondamenti di Informatica T-1, 2018/2019 – Modulo 2

Prova d'Esame 2A di Giovedì 24 Gennaio 2019 – tempo a disposizione 2h

"cassa.txt":

```
19 primo frutta contorno @@@ 23 secondo frutta contorno
19 secondo secondo contorno
45 secondo primo @@@ 78 contorno @@@ 12 primo frutta
45 secondo contorno contorno
```

"tariffe.txt":

```
secondo 2.35
primo 2.05
secondo 2.20
contorno 0.60
frutta 0.50
dolce 3.70
dolce 3.80
primo 2.15
secondo 2.30
```